6
Operações em um sistema de conferência

Para validação e teste do Modelo de Operações e de sua implementação no ambiente HyperDE foi desenvolvida uma aplicação de exemplo baseada em submissão de artefatos para uma conferência. Esses artefatos podem ser de três tipos: artigos, palestras ou *workshops*. A aplicação apresenta quatro tipos de atores:

- Autor: produz o artefato e o submete a conferência;
- Avaliador: avalia o artefato;
- *Chair*: define quais avaliadores irão avaliar quais artefatos;
- Administrador: desempenha algumas tarefas de administração do sistema de conferência.

Neste Capítulo detalharemos apenas as tarefas que contemplam operações. Inicialmente, apresentamos os UIDs referentes a essas tarefas e os Cartões de Operação correspondentes. As operações foram identificadas a partir dos UIDs, por meio da utilização das quatro diretrizes descritas no Capítulo 4. Um resumo das diretrizes é mostrado abaixo:

- Diretriz 1: uma operação deve ser criada para cada opção anexada a uma transição de estado que conectar uma ou mais estruturas ou um ou mais itens de dados a um estado de interação de destino.
- Diretriz 2: uma operação deve ser criada para cada opção anexada a uma transição de estado que conectar todo um estado de interação de origem a outro estado de interação de destino.
- Diretriz 3: uma operação deve ser criada para cada opção anexada a uma transição de estado que conectar uma estrutura ou item de dados ao mesmo estado de interação.
- Diretriz 4: uma operação deve ser criada quando houver uma opção anexada a uma transição de estado para conectar todo um estado de interação a ele mesmo.

Será explicitado em cada UID, qual foi a diretriz utilizada.

6.1.
UIDs e Cartões de Operação

- **UID 1: Fazer cadastro**

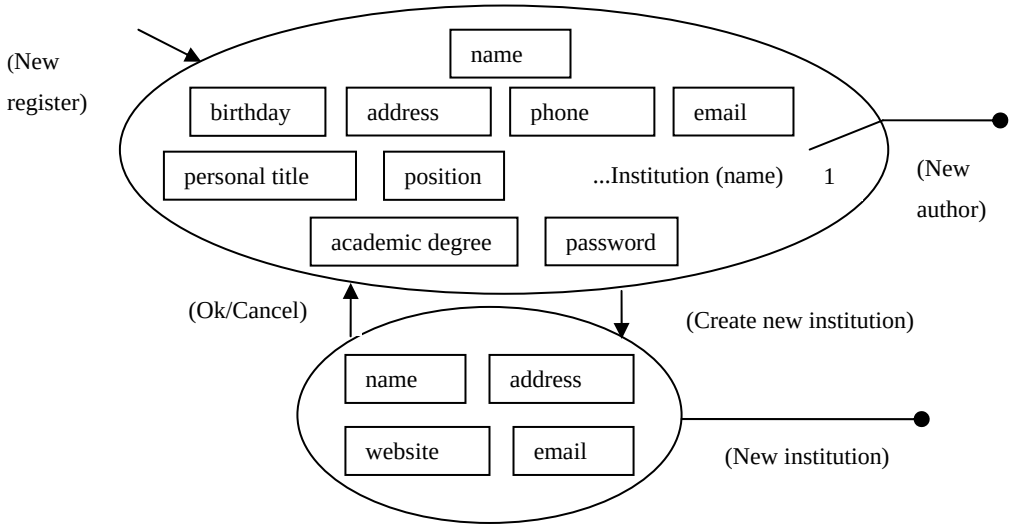


Figura 34 – UID 1 – Fazer Cadastro

Operation: new_author	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
¬ ∃ author(Author(author))	∃ author(Author(author)) execute Print_Inclusion_Error()
Parameter In: name:String birthday:Date address:String phone:String position:String title:String academic_degree:String email:String password:String login:String institution:Institution	
Parameter Out:	
Postcondition: new(author, Author)	

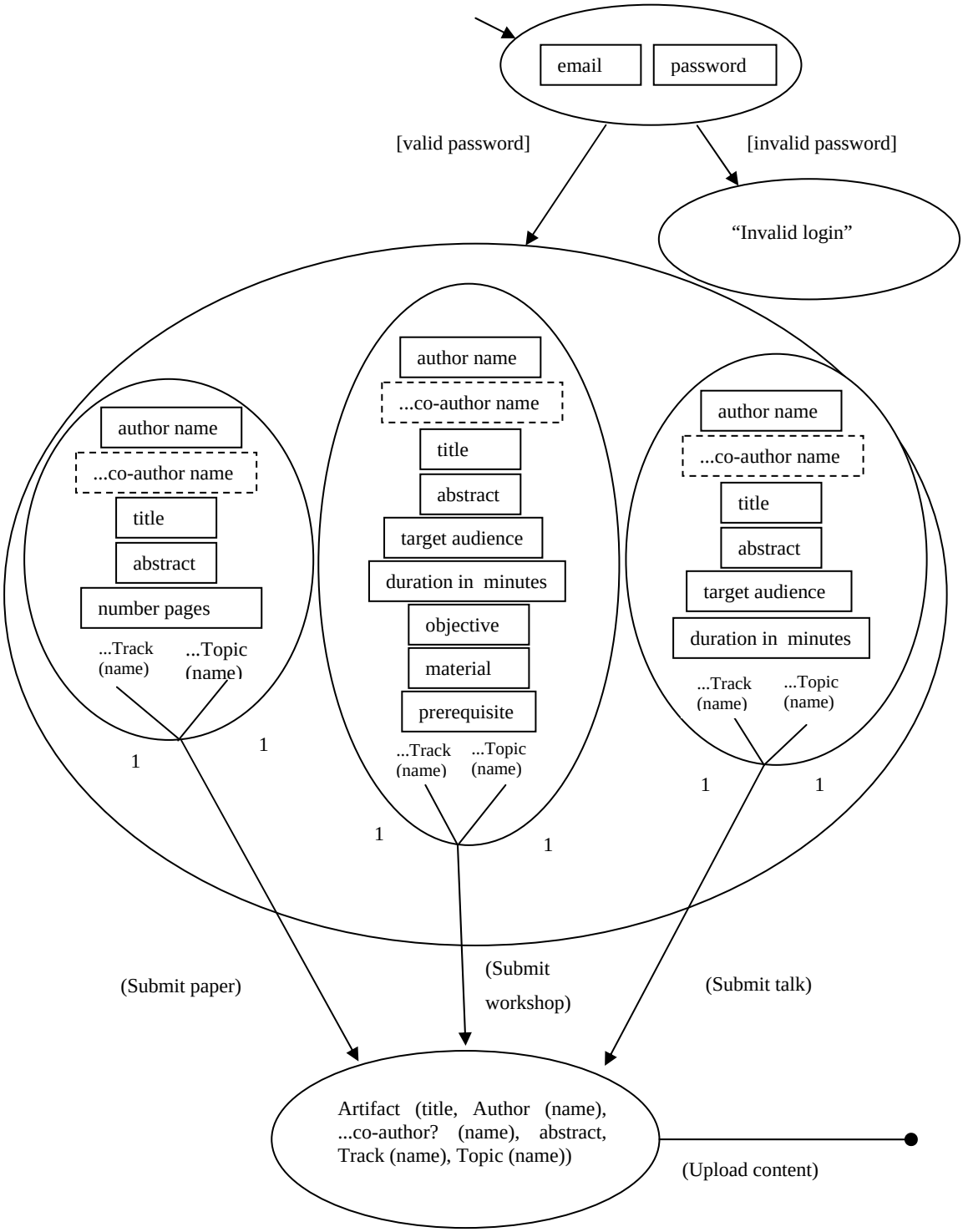
Figura 35 – Cartão de Operação de “new_author”

Operation: new_institution	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
Parameter In: name:String address:String website:String email:String	
Parameter Out:	
Postcondition: new(institution, Institution)	

Figura 36 – Cartão de Operação de “new_institution”

No UID 1 foram identificadas duas operações: “New author” e “New institution”. Foi utilizada a diretriz 4 neste UID.

- **UID 2: Registrar artefato (artigo, palestra, workshop)**



Precondition: user should be logged in as Author and if the submission final date expires, the application should not allow any artifact's submission.

Figura 37 – UID 2 – Registrar artefato

Operation: submit_paper	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Author))	¬type(user(Author)) execute Print_UserType_Error()
∀ submissionPaperDate (submissionPaperDate<submissionFinalDate)	∃ submissionPaperDate (submissionPaperDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In:	
author:Author coauthor:String[0..N] title:String abstract:String number_pages:Integer track:Track topic:Topic	
Parameter Out:	
Postcondition:	
new(paper, Paper)	

Figura 38 – Cartão de Operação de “submit_paper”

Operation: submit_talk	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Author))	¬type(user(Author)) execute Print_UserType_Error()
∀ submissionTalkDate (submissionTalkDate<submissionFinalDate)	∃ submissionTalkDate (submissionTalkDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In:	
author:Author coauthor:String[0..N] title:String abstract:String target_audience:String duration_in_minutes:Integer track:Track topic:Topic	
Parameter Out:	
Postcondition:	
new(talk, Talk)	

Figura 39 – Cartão de Operação de “submit_talk”

Operation: submit_workshop	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Author))	¬type(user(Author)) execute Print_UserType_Error()
∀ submissionWorkshopDate (submissionWorkshopDate<submissionFinalDate)	∃ submissionWorkshopDate (submissionWorkshopDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In:	
author:Author coauthor:String[0..N] title:String abstract:String target_audience:String duration_in_minutes:Integer objective:String material:String prerequisite:String track:Track topic:Topic	
Parameter Out:	
Postcondition:	
new(workshop, Workshop)	

Figura 40 – Cartão de Operação de “submit_workshop”

Operation: upload_content	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Author))	¬type(user(Author)) execute Print_UserType_Error()
∀ submissionArtifactDate (submissionArtifactDate<submissionFinalDate)	∃ submissionArtifactDate (submissionArtifactDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In:	
artifact:Artifact content:PDF	
Parameter Out:	
Postcondition:	

Figura 41 – Cartão de Operação de “upload_content”

No UID 2 foram identificadas quatro operações: “Submit paper”, “Submit workshop”, “Submit talk” e “Upload content”. Para as três primeiras operações foi utilizada a diretriz 2. Para a quarta operação foi utilizada a diretriz 4.

- **UID 3: Modificar informações de um artefato registrado (artigo, palestra e workshop)**

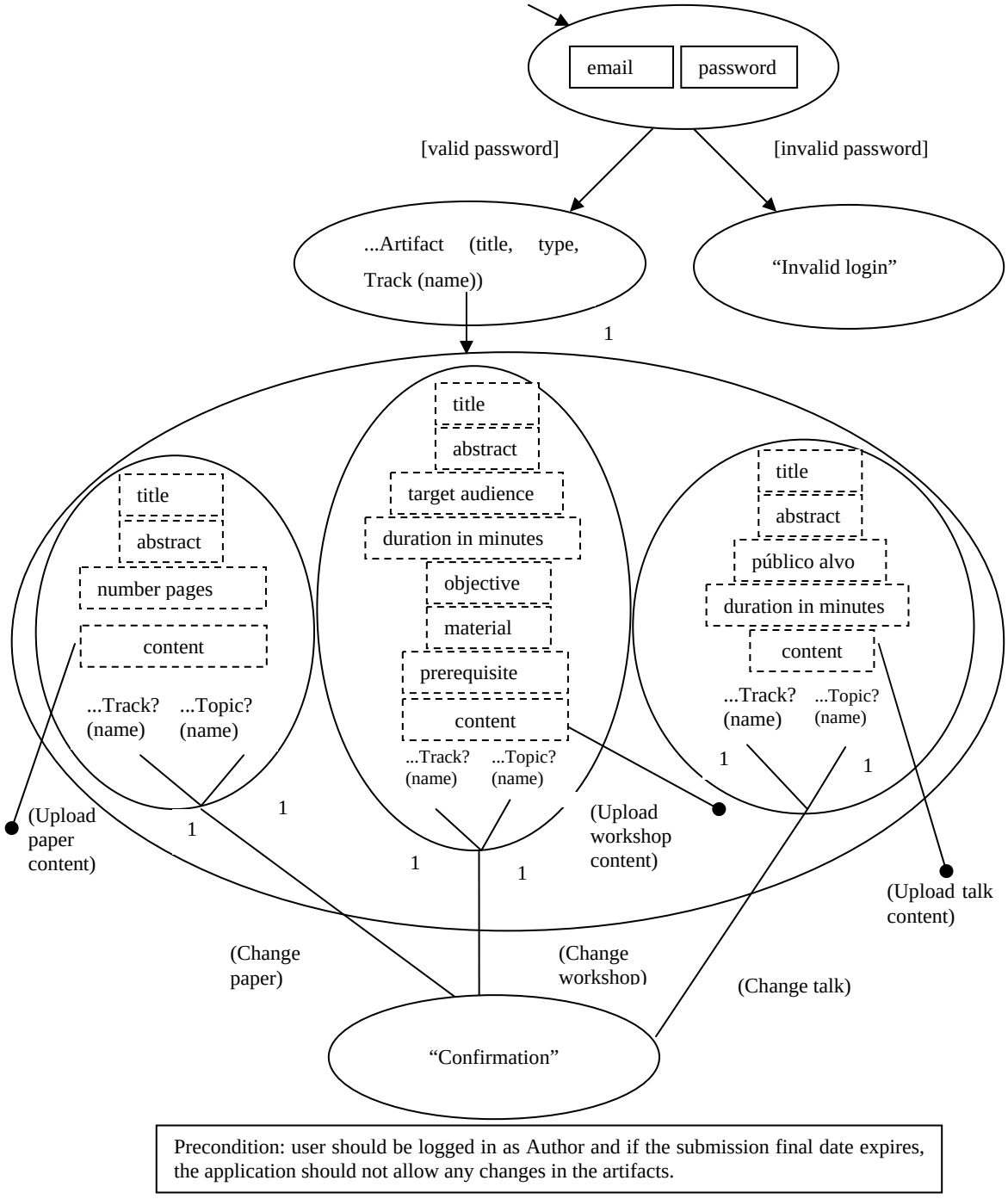


Figura 42 – UID 3 – Modificar informações de um artefato registrado

Operation: upload_paper_content	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Author)) ∀ submissionPaperDate (submissionPaperDate<submissionFinalDate)	**Failure_Handling:** ¬type(user(Author)) execute Print_UserType_Error() ∃ submissionPaperDate (submissionPaperDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In: paper:Paper content:PDF[0..1]	
Parameter Out:	
Postcondition:	

Figura 43 – Cartão de Operação de “upload_paper_content”

Operation: upload_workshop_content	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Author)) ∀ submissionWorkshopDate (submissionWorkshopDate<submissionFinalDate)	**Failure_Handling:** ¬type(user(Author)) execute Print_UserType_Error() ∃ submissionWorkshopDate (submissionWorkshopDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In: workshop:Workshop content:PDF[0..1]	
Parameter Out:	
Postcondition:	

Figura 44 – Cartão de Operação de “upload_workshop_content”

Operation: upload_talk_content	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Author))	¬type(user(Author)) execute Print_UserType_Error()
∀ submissionTalkDate (submissionTalkDate<submissionFinalDate)	∃ submissionTalkDate (submissionTalkDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In:	
workshop:Workshop	
content:PDF[0..1]	
Parameter Out:	
Postcondition:	

Figura 45 – Cartão de Operação de “upload_talk_content”

Operation: change_paper	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Author))	¬type(user(Author)) execute Print_UserType_Error()
∀ submissionPaperDate (submissionPaperDate<submissionFinalDate)	∃ submissionPaperDate (submissionPaperDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In:	
paper:Paper	
title:String[0..1]	
abstract:String[0..1]	
number_page:Integer[0..1]	
track:Track[0..1]	
topic:Topic[0..1]	
Parameter Out:	
Postcondition:	

Figura 46 – Cartão de Operação de “change_paper”

Operation: change_workshop	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Author))	¬type(user(Author)) execute Print_UserType_Error()
∀ submissionWorkshopDate (submissionWorkshopDate<submissionFinalDate)	∃ submissionWorkshopDate (submissionWorkshopDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In:	
workshop:Workshop	
title:String[0..1]	
abstract:String[0..1]	
target_audience:String[0..1]	
duration_in_minutes:Integer[0..1]	
objective:String[0..1]	
material:String[0..1]	
prerequisite:String[0..1]	
track:Track[0..1]	
topic:Topic[0..1]	
Parameter Out:	
Postcondition:	

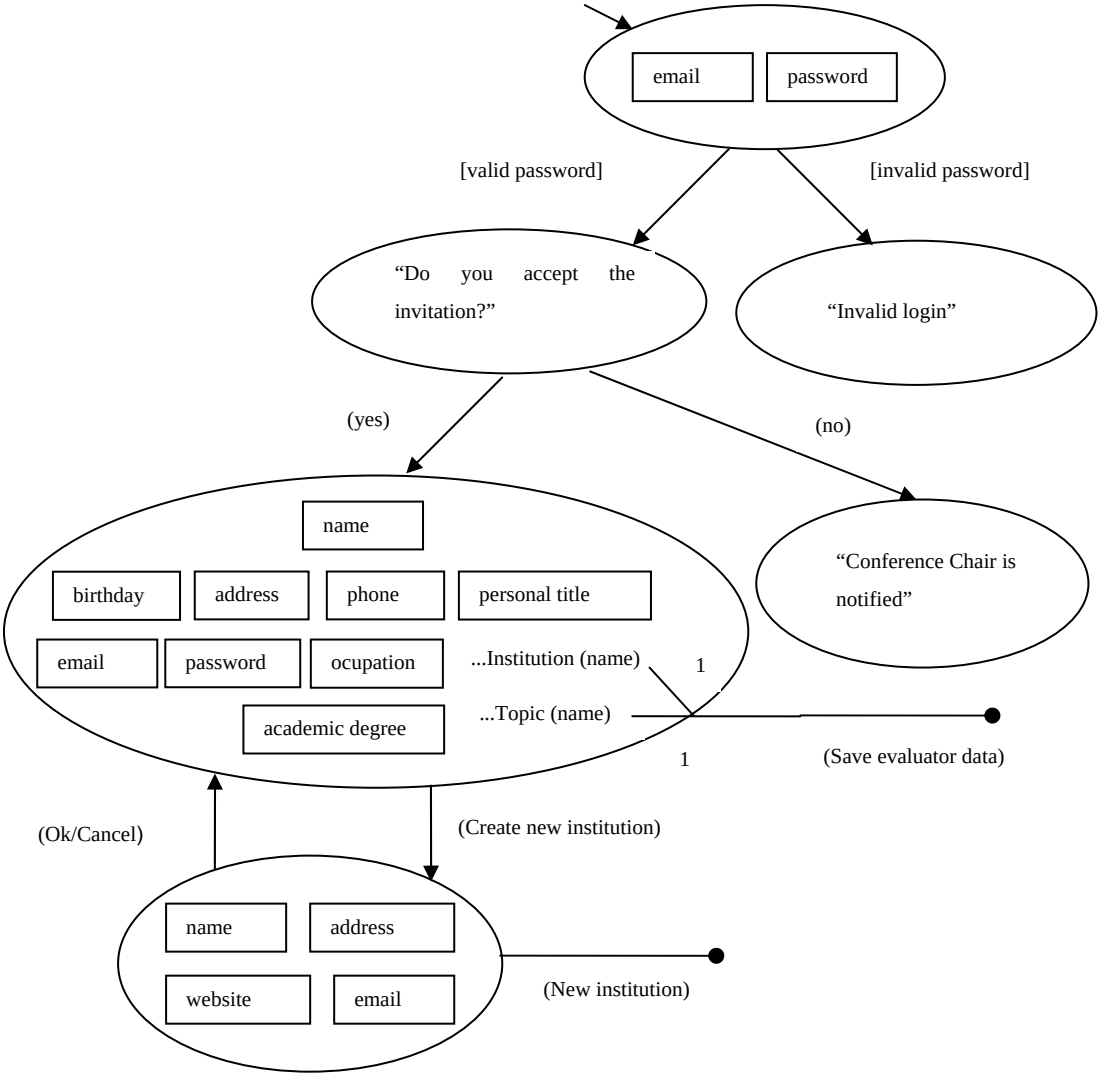
Figura 47 – Cartão de Operação de “change_workshop”

Operation: change_talk	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Author))	¬type(user(Author)) execute Print_UserType_Error()
∀ submissionTalkDate (submissionTalkDate<submissionFinalDate)	∃ submissionTalkDate (submissionTalkDate>submissionFinalDate) execute Print_SubmissionDateType_Error()
Parameter In:	
talk:Talk	
title:String[0..1]	
abstract:String[0..1]	
target_audience:String[0..1]	
duration_in_minutes:Integer[0..1]	
track:Track[0..1]	
topic:Topic[0..1]	
Parameter Out:	
Postcondition:	

Figura 48 – Cartão de Operação de “change_talk”

No UID 3 foram identificadas seis operações: “Upload paper content”, “Upload workshop content“ , “Upload talk content”, “Change paper”, “Change workshop” e “Change talk”. Para as três primeiras operações foi utilizada a diretriz 3. Para as três últimas operações foi utilizada a diretriz 2.

- **UID 7: Aceitar/rejeitar convite para avaliação do artefato (artigo, palestra e workshop)**



Precondition: user should be logged in as Evaluator and if the Evaluator doesn’t accept or rejects the invitation made by the Conference Chair after 10 days, the application sends a notification to the Conference Chair and cancels the evaluator’s invitation.

Figura 49 – UID 7 – Aceitar/rejeitar convite para avaliação do artefato

Operation: save_evaluator_data	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Evaluator))	¬type(user(Evaluator)) execute Print_UserType_Error()
∀ invitation	∃ invitation (Invitation(¬notExpired V ¬accepted))
(Invitation(notExpired Λ accepted))	execute Print_Invitation_Error()
Parameter In:	
user:Evaluator	
name:String	
birthday:Date	
address:String	
phone:String	
position:String	
title:String	
academic_degree:String	
email:String	
password:String	
login:String	
institution:Institution	
topic:Topic	
Parameter Out:	
Postcondition:	
new(evaluator, Evaluator)	

Figura 50 – Cartão de Operação de “save_evaluator_data”

No UID 7 foram identificadas duas operações: “Save evaluator data” e “New institution”. Para as duas operações foi utilizada a diretriz 4. A operação “New institution” não possui Cartão de Operação neste UID, pois já foi detalhada no UID 1.

- **UID 8: Avaliar artefato (artigo, palestra e workshop)**

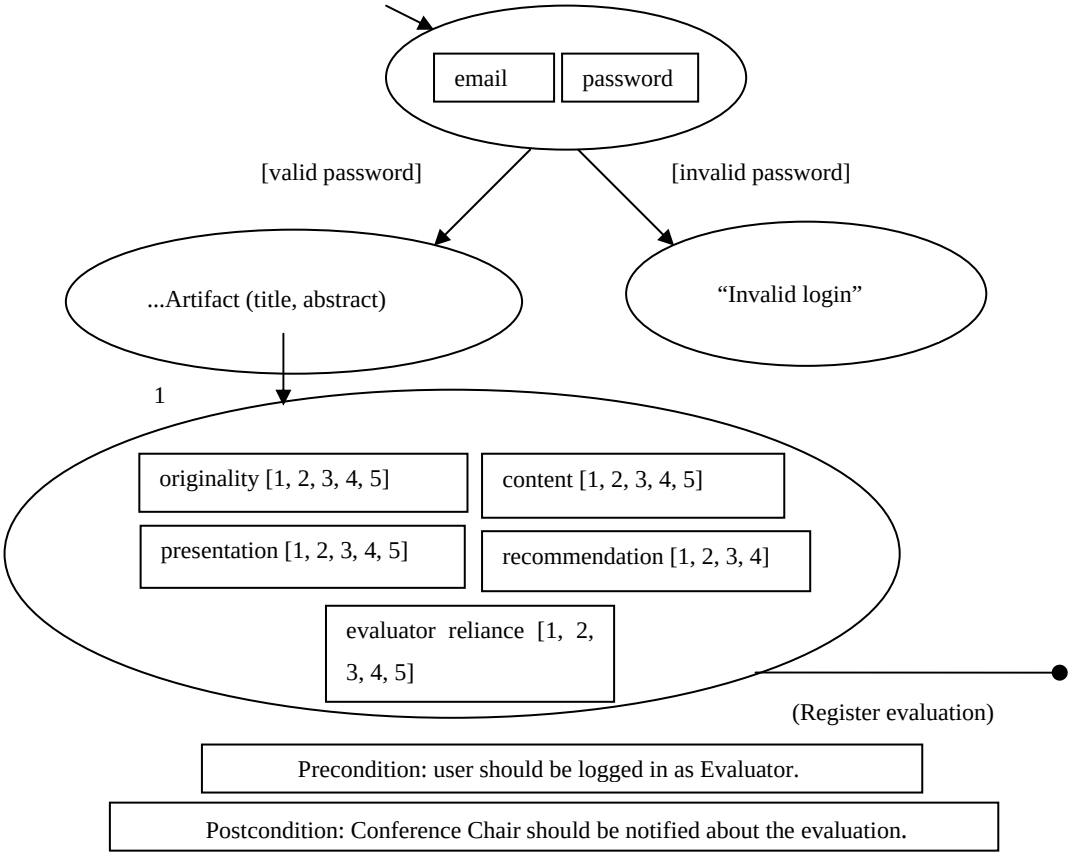


Figura 51 – UID 8 – Avaliar artefato

Operation: register_evaluation	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Evaluator))	**Failure_Handling:** ¬type(user(Evaluator)) execute Print_UserType_Error()
Parameter In: artifact:Artifact evaluator:Evaluator originality:Integer content:Integer presentation:Integer recommendation:Integer evaluator_reliance:Integer	
Parameter Out:	
Postcondition: send(notifiedEvaluation, Chair)	

Figura 52 – Cartão de Operação de "register_evaluation"

No UID 8 foi identificada uma operação: “Register evaluation”. Para esta operação foi utilizada a diretriz 4.

- **UID 11: Definir preferência de avaliação para cada artefato (artigo, palestra e workshop) a ser avaliado**

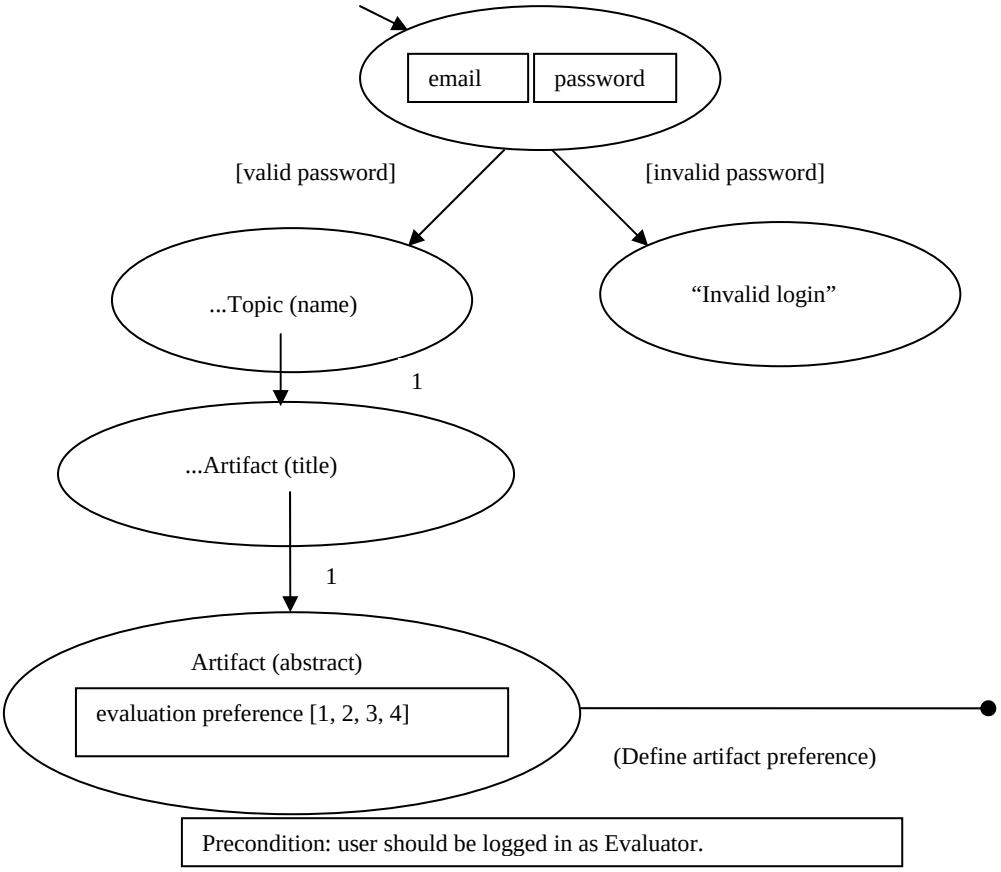


Figura 53 – UID 11 – Definir preferência de avaliação para cada artefato

Operation: define_artifact_preference	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Evaluator))	**Failure_Handling:** ¬type(user(Evaluator)) execute Print_UserType_Error()
Parameter In: artifact:Artifact evaluator:Evaluator evaluation_preference:Integer	
Parameter Out:	
Postcondition:	

Figura 54 – Cartão de Operação de “define_artifact_preference”

No UID 11 foi identificada uma operação: “Define artifact preference”. Para esta operação foi utilizada a diretriz 4.

- **UID 14: Alterar cadastro do Chair**

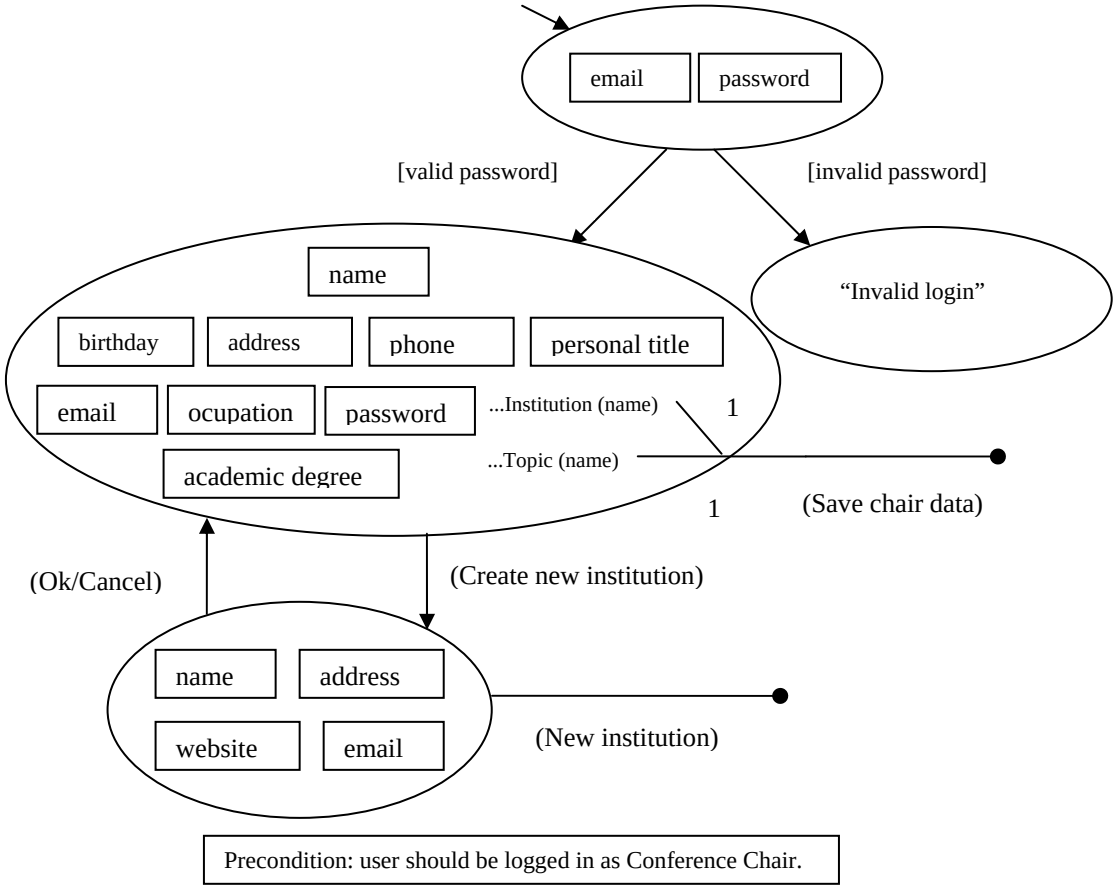


Figura 55 – UID 14 – Alterar cadastro do Chair

Operation: save_chair_data	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Chair))	¬type(user(Chair)) execute Print_UserType_Error()
Parameter In: user:Chair name:String birthday:Date address:String phone:String position:String title:String academic_degree:String email:String password:String login:String institution:Institution topic:Topic	
Parameter Out:	
Postcondition: new(chair, Chair)	

Figura 56 – Cartão de Operação de “save_chair_data”

No UID 14 foram identificadas duas operações: “Save chair data” e “New institution”. Para as duas operações foi utilizada a diretriz 4. A operação “New institution” não possui Cartão de Operação neste UID, pois já foi detalhada no UID 1.

- **UID 15: Aceitar/Rejeitar/Recomendar para pôster o artefato (artigo, palestra e workshop)**

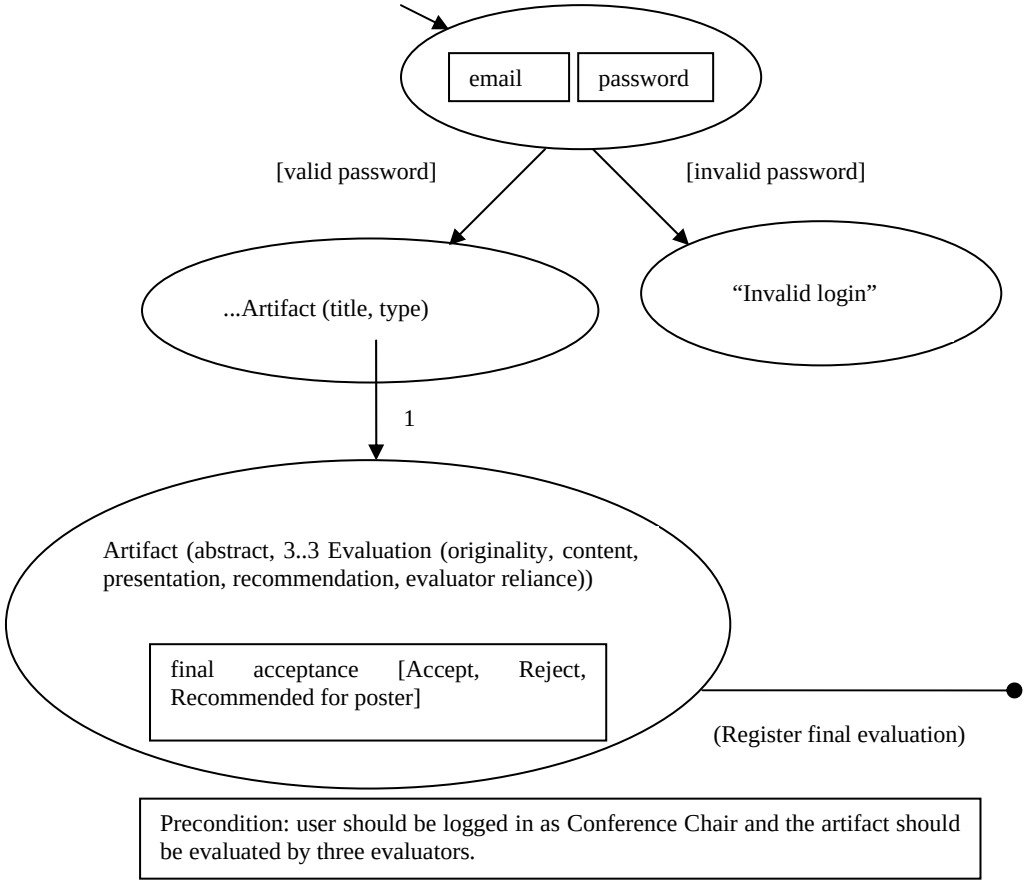


Figura 57 – UID 15 – Aceitar/Rejeitar/Recomendar para pôster o artefato

Operation: register_final_evaluation	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Chair))	¬type(user(Chair)) execute Print_UserType_Error()
∀ artifact(evaluation_quantity = 3)	∃ artifact(evaluation_quantity != 3) execute Print_EvaluationQuantity_Error()
Parameter In: artifact:Artifact chair:Chair final_acceptance:String	
Parameter Out:	
Postcondition:	

Figura 58 – Cartão de Operação de "register_final_evaluation"

No UID 15 foi identificada uma operação: “Register final evaluation”. Para esta operação foi utilizada a diretriz 4.

- **UID 16: Alocar avaliadores para o artefato (artigo, palestra e workshop)**

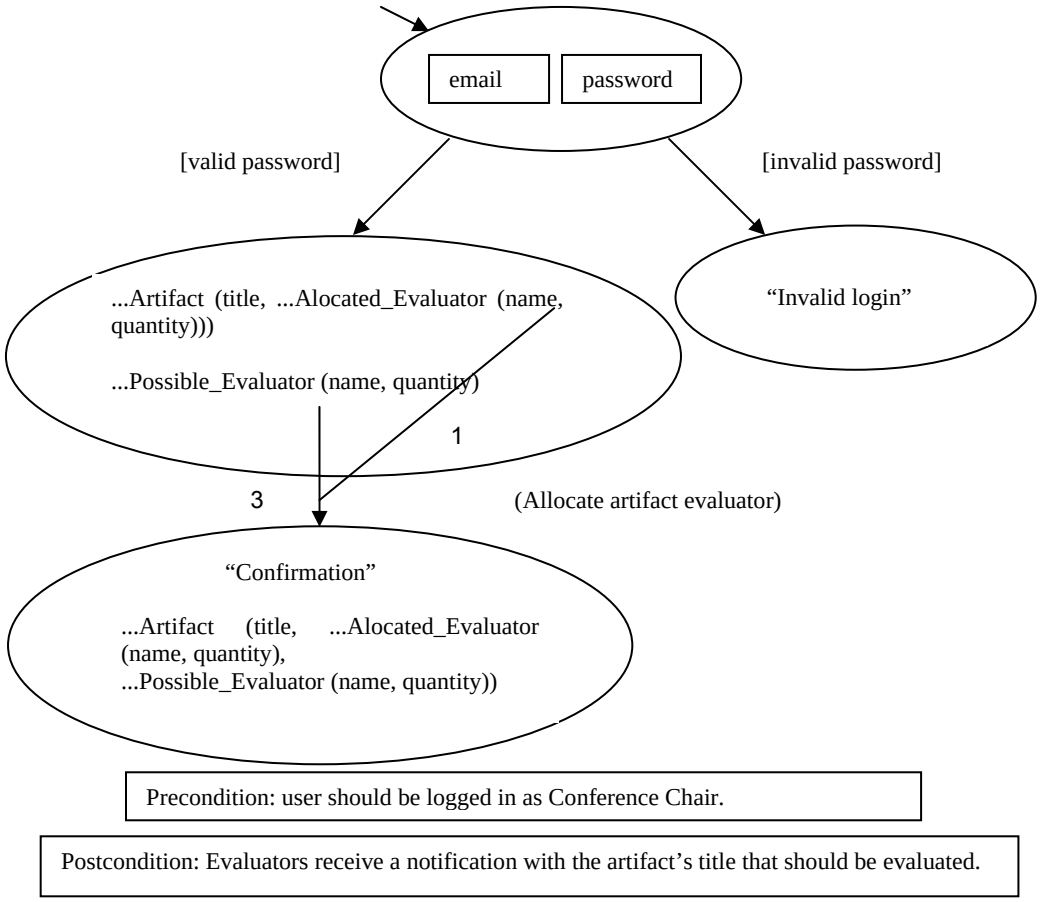


Figura 59 – UID 16 – Alocar avaliadores para o artefato

Operation: allocate_artifact_evaluator	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Chair))	**Failure_Handling:** ¬type(user(Chair)) execute Print_UserType_Error()
Parameter In: artifact:Artifact evaluator:Evaluator[3..3]	
Parameter Out:	
Postcondition: send(notification(Artifact(title)), Evaluator)	

Figura 60 – Cartão de Operação de “allocate_artifact_evaluator”

No UID 16 foi identificada uma operação: “Allocate artifact evaluator”. Para esta operação foi utilizada a diretriz 1.

- **UID 17: Criar trilhas**

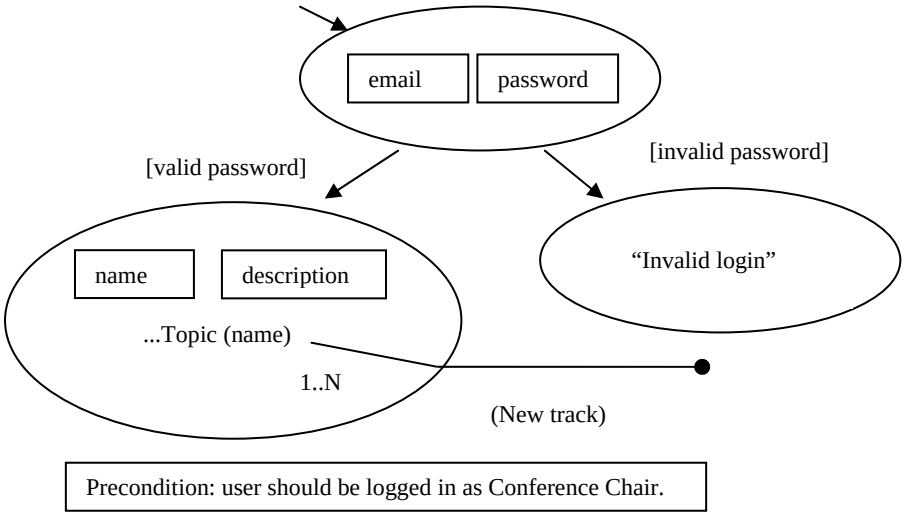


Figura 61 – UID 17 – Criar trilhas

Operation: new_track	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Chair))	**Failure_Handling:** ¬type(user(Chair)) execute Print_UserType_Error()
Parameter In: name:String description:String topic:Topic[1..N]	
Parameter Out:	
Postcondition:	

Figura 62 – Cartão de Operação de “new_track”

No UID 17 foi identificada uma operação: “New track”. Para esta operação foi utilizada a diretriz 4.

- **UID 18: Criar temas**

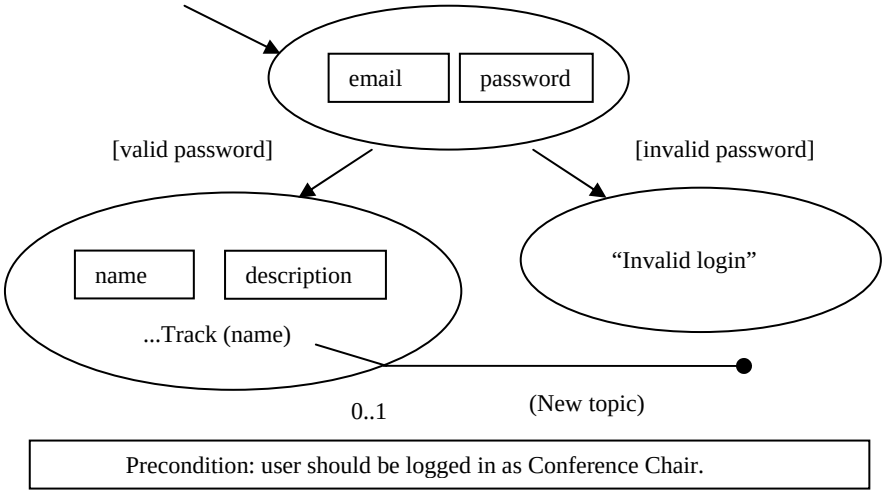


Figura 63 – UID 18 – Criar temas

Operation: new_topic	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Chair))	**Failure_Handling:** ¬type(user(Chair)) execute Print_UserType_Error()
Parameter In: name:String description:String track:Track[0..1]	
Parameter Out:	
Postcondition:	

Figura 64 – Cartão de Operação de "new_topic"

No UID 18 foi identificada uma operação: "New topic". Para esta operação foi utilizada a diretriz 4.

- **UID 19: Enviar convite para avaliador**

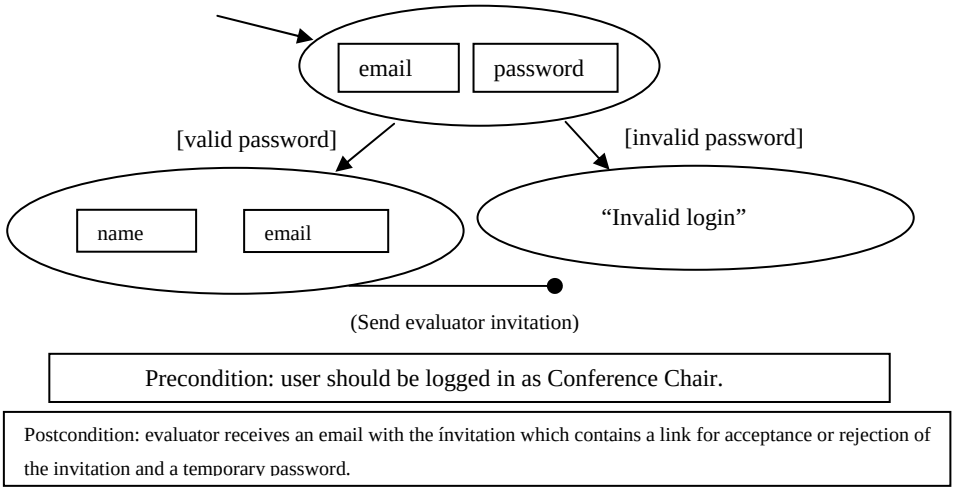


Figura 65 – UID 19 – Enviar convite para avaliador

Operation: send_evaluator_invitation	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Chair))	**Failure_Handling:** ¬type(user(Chair)) execute Print_UserType_Error()
Parameter In: name:String email:String	
Parameter Out:	
Postcondition: send(notifiedMail (link, password), Evaluator)	

Figura 66 – Cartão de Operação de "send_invitation_evaluator"

No UID 19 foi identificada uma operação: "Send evaluator invitation". Para esta operação foi utilizada a diretriz 4.

- **UID 21: Definir conflito de interesse para cada artefato (artigo, palestra e workshop) a ser avaliado**

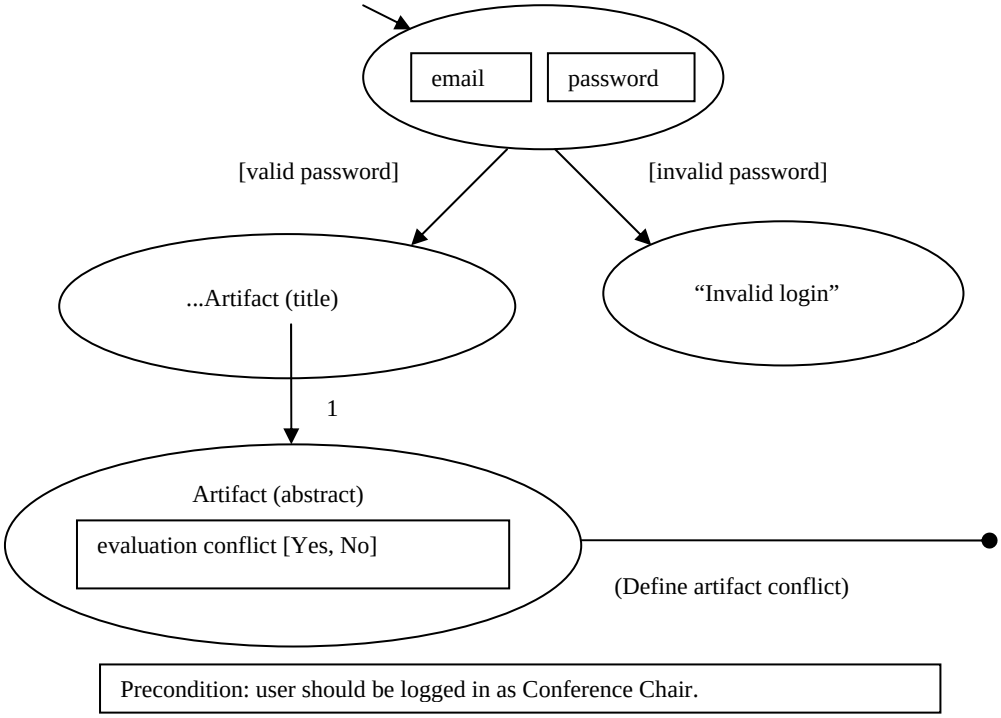


Figura 67 – UID 21 – Definir conflito de interesse para cada artefato

Operation: define_artifact_conflict	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Evaluator))	**Failure_Handling:** ¬type(user(Evaluator)) execute Print_UserType_Error()
Parameter In: artifact:Artifact evaluator:Evaluator evaluation_conflict:Boolean	
Parameter Out:	
Postcondition:	

Figura 68 – Cartão de Operação de "define_artifact_conflict"

No UID 21 foi identificada uma operação: "Define artifact conflict". Para esta operação foi utilizada a diretriz 4.

- **UID 29: Criar edição da conferência**

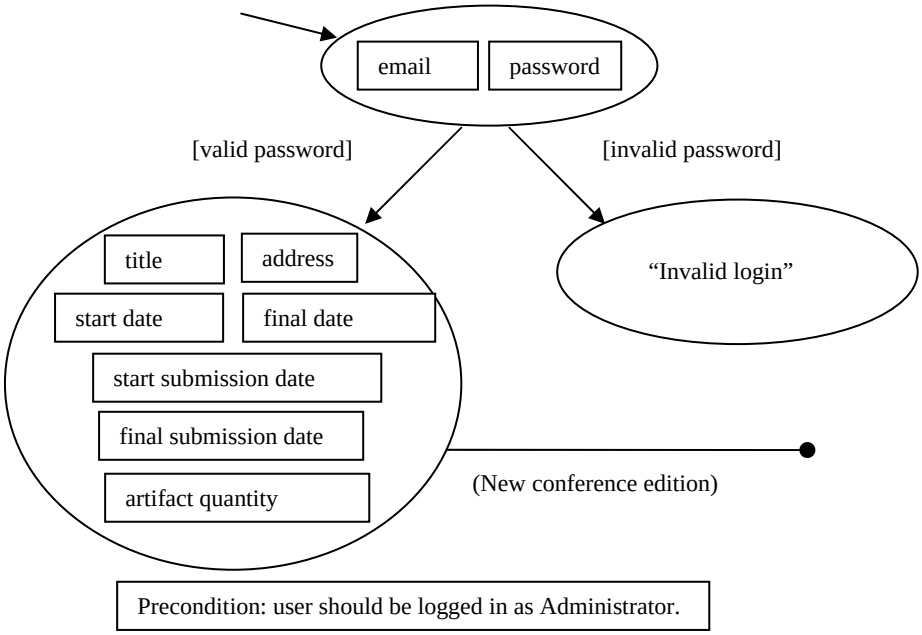


Figura 69 – UID 29 – Criar edição da conferência

Operation: new_conference_edition	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Administrator))	¬type(user(Administrator)) execute Print_UserType_Error()
Parameter In: title:String address:String startDate:Date finalDate:Date startSubmissionDate:Date finalSubmissionDate:Date artifact_quantity:Integer	
Parameter Out:	
Postcondition:	

Figura 70 – Cartão de Operação de "new_conference_edition"

No UID 29 foi identificada uma operação: "New conference edition". Para esta operação foi utilizada a diretriz 4.

- **UID 30: Criar Chair da Conferência**

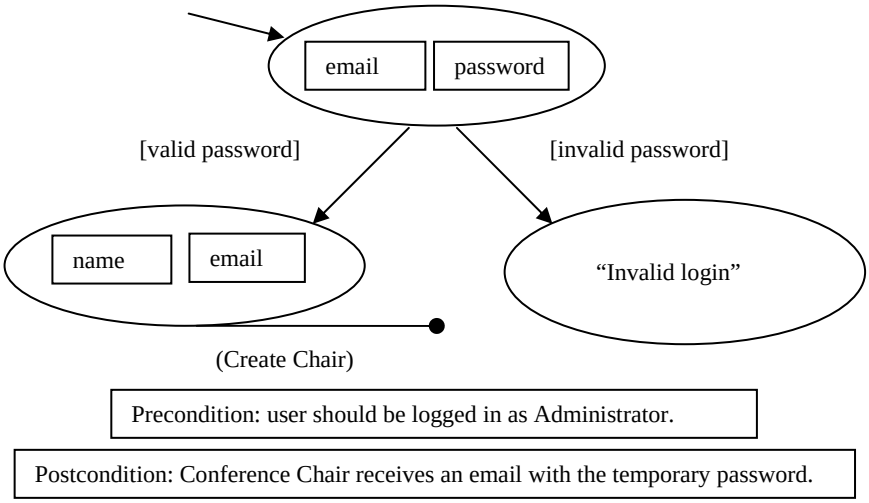


Figura 71 – UID 30 – Criar Chair da conferência

Operation: create_chair	
Language: Ruby	**Type:** internal_operation
Precondition: type(user(Administrator))	**Failure_Handling:** ¬type(user(Administrator)) execute Print_UserType_Error()
Parameter In: name:String email:String	
Parameter Out:	
Postcondition: send(notifiedMail(password), Chair)	

Figura 72 – Cartão de Operação de "create_chair"

No UID 30 foi identificada uma operação: "Create chair". Para esta operação foi utilizada a diretriz 4.